Paper received: 12.05.2019. ***Journal of Agronomy, Technology and Engineering Management***
Paper accepted: 09.08.2019. *ISSN: 2620-1755* *Vladisavljević et al., 2019. Vol. 2 SI(1): 1-7*

Review paper

TRANSITION OF DOCUMENTATION MANAGEMENT PARADIGMS IN SOFTWARE PROJECTS INFLUENCED BY AGILE METHODOLOGIES

Radovan Vladisavljević[1,*], Dragan Soleša[1], Vladimir Šimović[2]

[1] University Business Academy in Novi Sad, Faculty of Economics and Engineering Management in Novi Sad, Cvećarska 2, 21000 Novi Sad, Serbia
[2] The University of Applied Sciences Baltazar Zaprešić, Vladimira Novaka 23, 10290 Zapresic, Croatia

*Corresponding author:
E-mail address: radovan.vladisavljevic@fimek.edu.rs

ABSTRACT: The scope of this paper is to present two different ways of managing documentation in the domain of software projects. Documentation is the foundation upon which projects are built, however, the creation of large-scale documentation is often an ineffective act that can jeopardize project progress. On the other hand, modern software development methodologies advocate a different approach that balances the scope and content of documentation. Throughout this paper, we will provide a comparative analysis that can help us better see the differences between the two approaches.

Key words: *software project management, software development methodologies, project documentation, agile methods*

INTRODUCTION

Historically, software development has long been under the influence of the engineering sciences, leading to the creation of software project management methodologies based on the wrong settings. From the very first projects, it was evident that the logic of project management from other branches of engineering is not so easily applicable in software development. Firstly, classical engineering logic dictates the examination, documentation of problems, and only then to take other actions that lead to the solution of complex engineering problems. Early software creation methodologies just followed this logic. However, it quickly turned out that this mindset was not effective enough.

Time spent meeting all user requirements can be better utilized, especially considering that for most business software development time is critical. While the team developing the new software learns and documents all aspects of the problem, there may be a change in starting assumptions.

On the other hand, the development of tools for software project management helps to automate document creation. Besides, modern software systems in project management make it easier to make radical changes.

PROJECT MANAGEMENT

There are many definitions of projects, such as (Avlijaš and Avlijaš, 2018) that project management is a focused effort that has a goal and is timely determined. In addition to the goal and time, other authors include (Jovanović, 2015) the uniqueness of the project as an important feature. According to the Project Management Institute (PMI), a project is a temporary endeavor that aims to create a unique product or service (PMI, 2017). All these definitions tell us that project management is a management technique aimed at solving highly complex problems.

This technique has become extremely popular over the last twenty years; however, it can hardly be adapted for process management. Process management is dominant in the domain of production and similar processes that are repeated over a long time project. This technique involves a large number of functions and processes.

Project management is increasingly viewed as a holistic process as it involves a large number of closely related elements. According to research (Aramo-Immonen and Vanharanta, 2009) related to a holistic approach indicates that key qualitative factors are related to human resources management, production integration and cooperation with partners.

This indicates that projects are extremely powerful as a tool and as a system that can solve highly complex problems. What is interesting is that in many cases projects are relatively independent of the parent company. The project management system itself essentially advocates the creation of parallel management structures.

When a company has multiple projects then it takes a different approach to project management. This is a case where a company has a large number of projects that can start drawing more resources at one time without delivering the desired results. It is also possible to initiate projects that are substantially overlapping in content and purpose. In such cases, it is best to manage projects with portfolio management. This technique integrates all projects intending to create a management system that eliminates potential problems.

According to La Bros (2010), the advantages of portfolio management are reflected in the following elements: portfolio project analysis sorts projects by proximity system with strategic goals of the organization, using this method resources are directed to priority projects, eliminating inefficient and outdated projects, crucial projects are monitored to maximize their performance.

SOFTWARE PROJECTS

Software creation is a complex, time-consuming and resource-constrained process. All this tells us that creation is best with a project approach. Fundamentally, software project management is a blend of skills and science focused on planning and executing software projects. (Stellman and Greene, 2006) It follows that successful completion of a software project requires detailed knowledge of each segment of the project lifecycle.
The importance of software in the modern business is growing, with the digital economy increasingly talked about today. The digital economy, also known as the Internet economy, is an economy based on online transactions, mainly in the field of e-commerce (Turban et al., 2018). It follows that software is now of strategic importance for an enterprise.

In spite of all the above, a relatively large number of software projects have been unsuccessful. For decades, we have been working to create a better and better framework to reduce risks when running software projects. Initially, software projects were conducted similarly to other engineering projects, this approach proved ineffective.

What makes the software project special is the strong focus on people. Software projects are mainly people-oriented (Tsui, 2004). This is fully expected, given that the software project is mainly the product of the intellect of the development team.

We can view software projects as a separate branch of management that focuses much more on creativity than on technical aspects. The development team must combine analytical and creative skills to find out what the customer needs (Stepanek, 2012). Very often, the needs that contracting authorities have been contradictory. It is in this segment that human potential is crucial. Through various tools and techniques, as well as through intensive work with software contractors, an appropriate compromise can be achieved.

TIME HORIZON

Software projects like all other projects have limited resources, scope and a defined timeframe within which the project must be delivered. Of all the resources, time is one of the most interesting; the time horizon is often one of the most vulnerable elements of software projects. Namely, time cannot be made up for and time loss often entails several activities that are detrimental to other elements of the project.

It is very often that management pressures the development team to start coding as soon as possible to shorten the development time of a new software product (Chemuturi, 2013). Ignoring the planning phase to shorten software development time can jeopardize the progress of the entire project due to system errors.

For this reason, the time horizon plays a large role in determining the further course of software project development. The nature of the project plays a large role in determining which value is most important. In some projects, time does not play a big role but that is why the resources or scope of the project are elements that are far more important. An example of this is the critical systems on which human lives depend, such as software that controls the flight of an aircraft or software that controls the operation of a medical device.

SOFTWARE DEVELOPMENT PARADIGMS

No matter what paradigm the development team uses to develop the software, we can generally define the following activities (Sommerville, 2016):

1. Software specification
2. Software development
3. Software validation
4. Software evolution

The biggest difference between the paradigms of software project management is how the separation of these activities is approached. For "classic" models of software product development, the activities mentioned are strictly separate, and subsequent activity cannot be initiated unless the previous one is completed.

Contrasting to this approach are models that can be categorized as "agile" methods; in these models, the activities mentioned are not so strictly separated. Besides, agile methods are not linear, but activities can take place in parallel. However, many agile models require that activities repeat all the time.

Due to the nature of the work, we will not go into a deeper analysis of the differences between older and new models of development. We will limit ourselves to the relationship to documentation in "classic" or "agile" software development methods. Otherwise, these methods are directly related to paradigms, that is, the way to approach problems. Older methods are related to engineering paradigms and those that come outside the domain of software engineering. On the other hand, we have new project management paradigms that have emerged in the field of software engineering.

DOCUMENT MANAGEMENT

Based on the above stated, it can be said that older software development systems were largely managed similarly to any other engineering project. One of the peculiarities of classic projects is the relationship to the problem, i.e. documentation. At first glance, the documentation and the problem of the project do not have much relationship, but without the technical documentation that describes the problem of the project in a highly formalized way, engineering projects cannot be successfully managed.

On the other hand, we have a "classic" model, which is conditioned by the linear flow of the project, in other words, until one activity is completed the other cannot start. This directly means that documentation must be completed and delivered as soon as possible for one activity to be closed and another to be started.

This is one of the main reasons that affect the content of the documentation. Namely, for older software development models, the documentation must be complete and detailed. On the other hand, fewer details in the documentation are often not minor, the reason is that many details evolve (Rüping, 2003).

Document management in the domain of agile methods is based on this fact. In short, it can be said that there is a duality between the two ways of managing documentation. As an illustration, we will take the time it takes to complete the documentation.

TIME IT TAKES TO CREATE DOCUMENTATION

The purpose of software project management is to form a new software product in a controlled and predictable manner. In the case of "classical" methods, documentation is usually in a highly formal format. For agile methods, on the other hand, documentation is usually written in a more natural language.

However, the biggest differences between these two paradigms lie in the timing in which documentation must be created and delivered. For classic methodologies,

documentation must be created as soon as possible. The reason for this is that each subsequent activity depends on the previous one.

On the other hand, agile methods advocate a completely different way of creating documentation based on "organic" project development. As the project progresses, so does the documentation. In other words, documentation for agile methods is an integral part of the software product creation process itself.

This duality of documentation creation principles can be described as ASAP (as soon as possible) and ALAP (as late as possible). This distinction between ASAP and ALAP approaches gives us the best introduction to a paradigm shift. According to the ALAP approach, documentation must be prepared as early as possible.

For the project to run smoothly according to the older methodologies, documentation must be submitted as soon as possible. Each subsequent phase is initiated by the documentation from the previous activity. Unfortunately, this puts unnecessary pressure on development teams to create documentation faster. This leads to the creation of errors that multiply during the project.

The way to solve this problem can go in two diametrically opposite directions depending on the climate in the organization. The first direction goes toward solving the problem by returning the work to its previous activity. This can be quite expensive as previous activities may be in the domain of another development team that may no longer be current. The other direction may be intended to hide the problem so that the problem can be transferred to another department as quickly as possible. This is especially evident in those companies that have a “shooting the messenger” policy, in which cases the project's progression can be unsustainable (Kayser, 2011).

On the other hand, with the development model based on "agile" methodologies where documentation is delivered according to the ALAP principle, the problems described above may very rarely occur. The focus of modern software development methodologies is on the software itself, in light of this, we can say that documentation in such cases does not play a big role.

Table 1. The comparison of older and modern software development methodologies versus documentation

	"Classic" methods	**Agile methods**
Format	Formal	"Organic" with less form
Language	Technical	Natural
Level of detail	High	Low
Responding to errors found	Depending on the organization's climate	Fast
Purpose	Moving from one stage to another	Improving communication
Target group	Development team	The development team and contracting authorities

According to newer software development methodologies, documentation should link the development team and contracting authorities. This is a powerful technique because

you get constant feedback, so you can expect more flexibility for the development team better to understand user requirements.

CONCLUSIONS

Throughout the paper, only some differences between older and newer software product development methodologies are presented. At first glance, the flexibility and ease with which it communicates with users in recent methodologies create the illusion of superiority of these methods. However, the truth is quite different. Older methods are still current in the field of critical system development as well as in the development of extremely expensive software solutions.

Customers of complex and expensive projects are often state organizations that have to justify the development costs over some time. "Classic" methodologies provide a clearer picture of development that may be a little more intuitive for decision-makers who have no background in the software development domain.

On the other hand, we have new methodologies that are far faster and more efficient in developing software that has a wider range of uses. However, by developing methodologies based on new software development paradigms, new models can be expected to emerge shortly that will successfully replace older ones.

Two more points that are important need to be included in this consideration, namely the spread of computer literacy and the development of new software solutions that will enable greater customer involvement. More and more people are using computers and various services, so it is to be expected that we will soon have more and more users who, in terms of knowledge and experience, exceed their current users. The generational shift effect must not be forgotten here, too, so we will soon have decision-makers who have extensive knowledge in the computer science field.

Web development will result in more quantity and quality of communications, and it is expected here that younger generations will be able to communicate more easily through various services. All this leads to greater involvement of the contracting authority in the development of the software product.

REFERENCES

ARAMO-IMMONEN, H. and VANHARANTA, H. (2009). Project management: The task of holistic systems thinking. *Human Factors and Ergonomics in Manufacturing,* **19(6):** 582–600.

AVLIJAŠ, R. and AVLIJAŠ, G. (2018). Upravljanje projektima. *Singidunum, Beograd.*

CHEMUTURI, M. (2013). Requirements Engineering and Management for Software Development Projects. *Springer, New York, USA.*

JOVANOVIĆ, P. (2015). Upravljanje projektima. *Visoka škola za projektni menadžment, Beograd.*

KAYSER, T. (2011). Six ingredients for collaborative partnerships. *Leader to Leader,* **61:** 48–55.

LaBROSSE, M. (2010). Project-portfolio management. *Employment Relations Today,* **37(2):** 75–79.

PMI (2017). Guide to the project management body of knowledge. Sixth Edition, *Project Management Institute, USA.*

RÜPING, A. (2003). Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects. *John Wiley & Sons, Chichester, UK.*

STELLMAN, A. and GREENE, J. (2006). Applied Software Project Management. O'Reilly: *Sebastopol, USA.*

STEPANEK, G. (2012). Sofware Project Secrerts. *Apress, USA.*

TSUI, F.F. (2004). Managing software projects. Jones and Bartlett Publishers. *Boston, USA.*

TURBAN, E., OUTLAND, J., KING, D., LEE, K.J., LIANG, T. and TURBAN, C.D. (2018). Electronic Commerce 2018: A Managerial and Social Networks Perspective. Ninth Edition, *Springer, Kihei, USA.*